XML as a MultiMedia Window into Science Data

Donald C. Brown
ECOlogic Corp.
ESTO Prototype ID. 59.0

Abstract

We are prototyping an XML based system to provide metadata services that mirror data services, such as subsetting, that are used to derive new products from HDF-EOS source products. The HDF-EOS API provides services on the data without providing corresponding services on all EOSDIS meta-data, which can result in new data products with invalid meta-data. This project focuses on the impacts that XML and related technologies can have on creating data product with consistent data and meta-data. We are addressing issues around the use of metadata in EOSDIS and larger Earth Science community and with the location of metadata in the HDF-EOS file itself. To focus the effort, data was selected from MODIS and MISR based on diverse subsetting criteria. Subsetting of data and the corresponding metadata has been used as the catalyst for identifying the issues that are pertinent to other services and data types. Our approach uses XML and XSL as tools to aid in configuring a framework that calls external applications to actually derive the new metadata values. An attempt to utilize existing subset tools and access data via the HDF-EOS and HDF APIs has made simultaneous services on data and metadata impossible at this juncture. Our approach, therefore, is to provide a tool to augment existing sub-setters by updating the new data products with consistent meta-data. Conclusions reached so far range on issues from the usability of XML Schema to impacts of the HDF-EOS metadata model on the ability to provide services to both data and metadata from the Terra project. Conclusions carry implications for future metadata models and the use of XML and related technologies for providing services for HDF-EOS data

1.0 Introduction

We are developing a prototype to investigate using eXtensible Markup Language (XML) and related technologies to provide a unified view of science data and metadata within a single document. Such a view of science data as multimedia might allow users to manage, manipulate and add value to existing self-describing science data formats. Applying XML as a window into a science data format such as the Hierarchical Data Format for the Earth Observing System (HDF-EOS) could support scenarios such as:

- Detailed commentary and analysis on granules by individual scientists could be stored directly with that granule.
- Services such as subsetting could operate on both metadata and data simultaneously
- Scientists could extend existing metadata standards to handle their unique metadata needs, allowing science teams outside EOS to participate in the generation of earth science products that the EOS systems could still ingest and manage.
- Science teams could productively use inexpensive and free generic XML software tools.

The prototype focuses on issues around providing services, such as subsetting and re-gridding, to the data. Investigation of existing subsetting services, HEW from the University of Alabama at Huntsville and the MISR subsetter from Langley Research Center, has indicated that changes were made to only part of the metadata and that the metadata that remained in the HDF-EOS subset file was partly reflective of the subsetted data and partly reflective of the parent, unsubsetted, data. For MISR data, at least, there is metadata that is held within HDF structures that are not accessible via the HDF-EOS interface and is left unchanged by the HDF-EOS subset routines. This leads to data type dependent subset tools. The HEW subsetter would not subset a MISR file and the MISR subset tool does not work on a MODIS file.

Service provision is data specific and is focused on the data and not the data and metadata.

In looking at how services can be provided to the data and metadata within the HDF-EOS file, it was clear that creation of an XML version of the HDF-EOS file would not offer much in terms of addressing the multiple issues that were present Storage of binary data within XML is an open issue and current approaches are not simply conducive to data manipulation. The algorithms needed to locate and subset data within an HDF-EOS file are part of the HDF-EOS API and would have to be re-implemented to work on XML. Existing subset tools would have to be re-written to work on XML documents rather than on HDF-EOS files. Over time, the amount of data that would have to be converted from HDF-EOS to XML would be extremely large. Clearly, benefits from XML and related technologies attained without conversion of HDF-EOS files to XML documents should be sought. Once the decision was made to use the existing APIs and tools to provide services to the data within the HDF-EOS files, the role of XML in providing services shifts from representation of the metadata and data in the file to representing the processing needed to provide services to the data and the metadata.

2.0 Problem Definition

2.1 Metadata Issues

HDF-EOS is a data format that extends Hierarchical Data Format (HDF) to provide specific data types that were needed to hold the data from Earth Observing System (EOS). It is an attempt to fit a wide variety of data formats into three basic structures. The use of basic structures provides some level of similarity of access of the data across different instruments and different platforms.

In order to accommodate HDF-EOS data requirements, data from some instruments made exceptional use of the HDF-EOS structures. MISR required a change in the original HDF-EOS Grid implementation in order to store swath-like products which are broken into equal size blocks.

The needs for the storage of metadata specific to the structures of the data did not fall neatly into the metadata structures that were available under HDF-EOS. HDF-EOS metadata is stored in three main structures: Structural, Core or Inventory, and Product or Archive metadata. Structural metadata is used by the HDF-EOS APIs to understand the structure of the data within the file, and the HDF-EOS APIs modify the structural metadata to reflect operations such as subsetting. Core metadata is used by ECS for search and retrieval of the data and is not altered by the HDF-EOS APIs during subset operatons. Core Metadataa can be operated on by the metadata tools that are part of the SDP Toolkit. Product metadata contains metadata that is important for the product, but which is not used in the searching and retrieval of the data.

Additional metadata is stored in a data specific manner. MISR metadata is broken into six classes, three of which are MISR specific. MISR allows for File, Grid/Swath, and block level metadata. Only Grid/Swath metadata is accessible via the HDF-EOS APIs. The other metadata have to be accessed via the HDF interface. This metadata is also left unchanged by the current subsetting tools.

The relationship between data and metadata within any specific HDF-EOS file is a function of the organization of data for that specific data type. A generic service tool requires the ability to change processing based on data type or the creation of a data representation which would abstract the specifics of many different data types. This prototype effort has focused on the first approach in an attempt to find a way to create operations that would work on both data and metadata concurrently.

2.2 Metadata – Data Consistency

Use of the data is the best indicator of whether the mismatch between subset data and the metadata contained in the file with it is a significant problem. Currently, data that is modified by a data services is seen as being re-createable and is not re-ingested into ECS or re-processed by the data centers. As the quantitiy of data grows and as specialized collections are

created, the ability to search on the metadata stored in a subsetted file will increase and the issues about data and metadata consistency will become critical. Either additional collection specific metadata will have to be added to allow for searching within a specific collection or adherence to a searchable metadata standard will have to be maintained.

The metadata held in the CoreMetadata reflects many of the issues around maintaining consistency of data and metadata. As part of the prototype effort, the parameters in the CoreMetadata have been classified as to their relationship with the modified data structures found in a derived product and as to their means of modification. For several parameters, such as QA flags, the methods for recalculating the values are unclear in terms of the location of the data and the algorithm needed for the recalculation.

One of the strengths of the HDF-EOS file format is that it creates a self-describing file. With the current state of service provision, the resulting files are HDF-EOS in name only since they are no longer truly self describing.

3.0 XML

XML, a subset of SGML (Standard Generalized Markup Language), was developed by W3C, the World Wide Web Consortium, as a result of a desire to provide a meta markup language to allow for more flexible tagging of data than was provided by HTML. HTML has a fixed set of tags which tie the display of data with the content of the data. XML separates the display of data from the structure and contents which allows for non-visual application usage of tags to find data and change processing on the data. XSL (eXtensible Style sheet Language) technoloy provides both a way of formatting the information in an XML document and providing a transformation language for the data in the XML document.

XML documents can be validated by using a Document Type Definition (DTD) file or by using an XML Schema. An XML Schema is itself an XML document that describes the structure of an XML document type including the types required for each of the elements in the document. DTDs are the older more stable technology; XML Schema is the new technology and has received W3 Recommendation status in May, 2001.

XML is text based and human readable to a point. It provides the ability to create new text tags that are interpreted by a corresponding style sheet or application. This allows display of text and images within a browser based on the browser's ability to parse the XML and associated style sheet. The XML document itself contains the structure of the data and allows for some relational information between the different structures. It is limited by only supporting a tree or hierarchical data structure.

While XML does a wonderful job of expanding the information that can be displayed on a website, it has also made inroads in the storage of information within a database. XML documents can be inserted directly into and retrieved from databases and search operations can be performed on the XML documents whether in a database or not.

XML also shares HTML's ability to indicate that data should be pulled from a different file. There is also an ability to pass instructions to an underlyinig application to guide the processing of the content of the XML document. This ability to permit variable processing of data is where our prototype has focused its efforts. Binary data, such as images, can be stored within the XML document itself in a 64 bit encoded text format, or read from an external file. Formatting and displaying the contents is done via XSL

4.0 Prototype Design

4.1 Design Drivers

The prototype design evolved based on several considerations. The initial plan to model HDF-EOS data with an XML representation raised more issues than it seemed to solve. First, access to the data in the HDF-EOS file has a well developed API that would have to be replicated in order to access the XML version of the data. This ruled out a simple XML representation of the HDF-EOS data. A concept of using the HDF-EOS APIs to create a new organization of the data raised different issues. The organization of the XML document would be different for different services. For example, organizing the data to support subset services efficiently would

be different than the organization needed for efficiently creating a mosaic of several different files. Beyond the organizational issues, there are issues with how the binary data in an HDF-EOS file would be maintained and interpreted since XML is a text based file type. Lastly, while the first two issues could be addressed with some potential chance of success, the actual provision of a service, e.g. subsetting, would have to be implemented then using the XML document to create a new XML document. At this point in the analysis, there did not seem to be a convincing argument that converting an HDF-EOS file to an XML format would provide much value.

Once the decision was made to not simply convert an HDF-EOS file to an XML document, then it was easy to assume the usage of the current tools for providing services. Interest was expressed by ESTO in development of an API that would allow developers of future services (e.g. regridding, creation of mosaics, etc.) create those services in a less data dependent manner. While such an API was addressed breifly in the original proposal, it was seen as an effort beyond the scope of the current protype effort. Still, answering questions about how such an API could be created remained one of the drivers in the design of the prototype.

The decision not to develop a separte subsetting tool made the prototype design dependent on the current subsettng tools. This in turn raises questions about the concept of providing services to the data and the metadata concurrently. While the situation in terms of current subset capabilities is intractable, answers have to be found for how to handle data and metadata concurrently for new services. This too became a design driver for the prototype.

In order to focus the efforts, the design has been limited to subsetting services on two ESDTs. There is a recognition, though, that the prototyp should be extensible to different data types and different services. The design has to indicate where there is a dependence on the data and on the type of service provided so that modifications to the design to accommodate different data and different services can be more easily identified. The limitation of the prototype effort to looking at different types of ESDT's means that the only file type addressed is HDF-EOS. While not explicitly stated as a driver for the prototype in the original statement of work, there has been an effort in the eveolution of the design to consider expansion of the prototype to additional data formats..

The design has also focused on the correction of the Core Metadata. This has been in large part due to the early focus on MODIS data which does not seem to have embedded metadata in HDF structures connected to the data objects. Review of the MISR Science Processing ICD indicates that the study of MISR data will expand the design of the prototype to handle metadata that is embedded in native HDF structures. There is also a desire to allow for additional metadata structures to be added to the HDF-EOS file by the scence user.

The last driver is the operations concept for the prototype. Early on, we decided on a GUI which would run the processing of the metadata for a subset file. This decision was based on a desire to give the initial users of the prototype control over the correction of the metadata and its re-insertion back into the HDF-EOS subset file. The current design is based on a GUI developed using Java Swing components. The advent of Java Server Pages makes the concept of running the prototype as a web application intriguing and will be addressed in the final report.

4.2 Components of the Prototype

Figure 4.2-*1 Components of the XML Metadata Correction Tool* shows the connection between the different components. The Java application and GUI control the extraction and correction of the metadata from the subset HDF-EOS file. GRUNK (GRammatical UNderstanding Kernel) is a product from the University of Illinois at Urbana-Champaign for creating XML from any other structured text based language (e.g. ODL) which is used to create XML documents from the ODL containing the Core Metadata. Given the focus of the initial design of the prototype on the issues with the Core Metadata, the prototype is currently designed to handle metadata that is held in ODL format only. Slight modifications to the design will be needed to handle metadata that is kept in HDF data structures (e.g. VDATA). XML-Spy, the third software component of the prototype is used for creation of XML Schema and XSL Style Sheets for each ESDT which will be supported by the prototype. The acutal

corrections to metadata values would be made via external code that is called a standalone applications (denoted by the C code component in the figure).

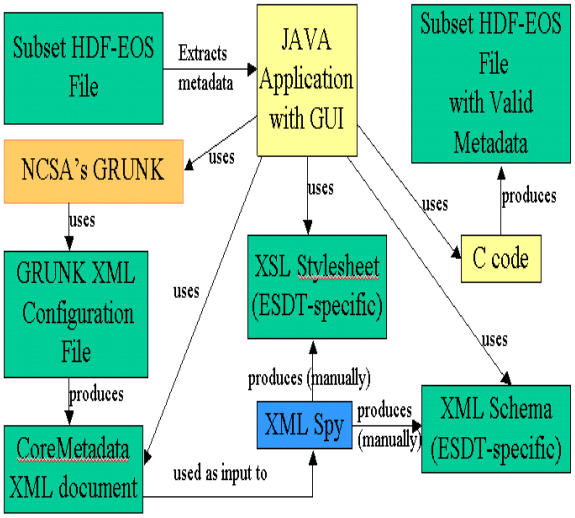

Figure 4.2-1 Components of the XML Metadata Correction Tool

The prototype requires several layers of configuration . There is a configuration file which is used for the general configuration of the GUI and Java application. It contains locations of supporting software (the external applications) and location of the log files. GRUNK requires its own configuration file. Both the application configuration file and the GRUNK configuration file are XML documents.

Each ESDT also requires configuration in the form of an XML Schema and two XSL style sheets. The XML Schema gives the generic structure of the metadata XML document for each ESDT and is used to validate the XML created from any given HDF-EOS file of that data type. One of the XSL style sheets is used for guiding the processing needed to correct the data values of the metadata parameters. The other XSL style sheet is used to convert the XML document with the corrected metadata values into an ODL document which can then be placed back into the HDF-EOS file.

4.3 Processing

The prototype has a very simple JAVA GUI that guides the user through a CoreMetadata corrections process. The GUI has one main screen with three buttons at the top that allow the user control of the entire CoreMetadata correction process. Only one of these three buttons will be sensitized at any given step in the process. In addition to the three buttons at the top of the screen, there are two text areas which allow the user to view the XML transformations produced in the uncorrected and corrected XML documents. There are also two buttons at the bottom of the screen which provide the user with access to information about the CoreMetadata conversion and replacement process, as well as the ability to restart the entire end-to-end process again. Figure 4.3-1 *Prototype Screen Layout as it Initially Appears,* shows the initial state of the GUI. The only main button available at the initial invocation of the tool is the *Select HDF File* button. Pushing the *Select HDF File* button allows the selection of an existing HDF-EOS file, which should be the product of one of the existing subset tools.

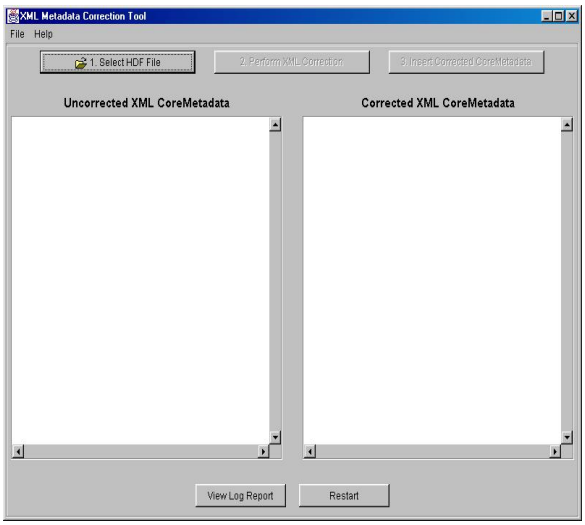

Figure 4.3-1 Prototype Screen Layout as It Initially Appears

The processing that follows will be focused on providing subset services to the metadata in the file based on the subset request that was performed on the data in the original HDF-EOS file. Tying together the provision of multiple services within the same application could be accomplished via several different operations concepts. Rather than spending prototype effort exploring those different operations scenarios,

the prototype makes the assumption of the type of service that is being performed and focuses only on subsetting.

Once the user has selected the file, the java application determines the ESDT of the file. The configuration parameters for the ESDT are retrieved. These parameters include the name and location of the XML Schema for the ESDT, the name and location of the XSL style sheets, location of the external applications that will be used for the correction of the metadata, and the name and location of the configuration file that will be used for the ODL to XML transformation.

The java application extracts the ODL for the Core Metadata from the file using an external application written in C. The application gets input from the command line arguments and writes the Core Metadata to standard out. The Java External Interface class reads the Core Metadata ODL from standard in and stores the ODL in memory. The Java interface used to call the application for getting the Core Metadata ODL creates a new process to execute the application with the arguments that are passed in to it. GRUNK is then called to change the ODL into an XML document that contains the uncorrected metadata. The XML document is validtaed against the XML Schema that exists for the ESDT and, if valid, displayed to the user in the *Uncorrected XML CoreMetadata* screen on the left hand side of the GUI. The button, *Perform XML Correction,* is then sensitized and the user can continue by hitting this button to start the correction of the XML CoreMetadata document.

When the user pushes the *Perform XML Correction* button, the GUI initiates the processing of the XML document containing the uncorrected metadata through the use of an ESDT specific XSL style sheet that will control the re-calculation of selected metadata parameters. The style sheet is parsed by the XSLT parser, Xalan, and embedded Java calls the Java External Interface passing the application to run and the input to the application. The external application returns the updated value as a string and the value is put into a new XML document. Metadata parameters that need no modification are simply copied from the original uncorrected XML CoreMetadata document into the new corrected XML document. The XML document with corrected metadata values is validated against the ESDT specific XML Schema. The C*orrected XML Core Metadata* text area is populated with the XML document created by the value correction process. Clicking on the *Insert Corrected CoreMetadata* button will cause the Java application, using another XSL style sheet for XML to ODL conversion, to create an ODL version of the corrected XML CoreMetadata document and write the ODL to the HDF-EOS file.

A dialog is displayed to indicate that the original CoreMetadata ODL file has been successfully replaced back into the HDF file, thus ending the end-to-end process. All main buttons are desensitized, the *Restart* button must be pushed.to re-initialize the tool. The origiinal subset file which had incorrect metadataa will now have metadata that is reflective of the data in the file.

5.0 Lessons Learned

The ways that metadata is held within the HDF-EOS file does not lend itself to providing services on both data and metadata. The location of the metadata and the relationships between the metadata and the data are specific to the data type that is being processed. XML with its ability to separate display and structure and its ability to aid in the transformation of data from one format to another seems to be a good fit for tyring to provide generic services to HDF-EOS files.

One approach would be to define generic format that would cover all the ESDTs and translate the individual ESDT to that format and have services provided on that format. Two problems have to be confronted. The first problem is the redevelopment of the APIs needed to provide services on the new format. This problem leads almost immediately to the second problem which is the definition of the data and metadata models for the new format. This is a duplication of the effort that gave rise to HDF-EOS as a format in the first place. The result would likely be an improvement in some areas and a degradation in others for providing services. The addition of yet another format would complicate the data management situation even more.

The approach of having a framework that will tie together processing based on data type specific configuration has some promise. It is possible that with some alterations, the current subset tools could be used by the framework. While the actual processing on data and metadata would not truly be simultaneous, from a user perspecitive they could be part of the same process. Additional subseting or other services could be added with changes made to the framework to suport them. Such changes would be connected to the need to have some logic in the framework to make sense of the data returned from the external applications. For example, the string based returns needed for subsetting metadata would be insufficient for subsetting the data itself. The applications executed by the framework could apply algorithms to both the data and the metadata.

The major difficulties are not so much technical as procedural. Configuration for each data type requires knowledge both of the processing needed for providing services to the metadata and the data as well as knowledge of XML Schema and XSL style sheets. Ideally, the science team would consider the requirements for providing services to the data and metadata as part of the creation of the data and metadata models. Either a parameter would be identified as not being modified for specific derived products or the method and/or algorithm for calculating the modified values would be identified as well as the source of the data for the re-calculation. Such an effort may not be a prioirity for the science team. This means that the knowledge of how to transform the data and metadata will happen after the data has been collected. Getting that information has been very difficult.

One of the goals of the original proposal for the prototype was the unification of data and metadata. In many ways, this is desireable since the distinction between data and metadata is context based. Within the confines of EOS data held in HDF-EOS formats, though, this does not seem to be an attainable goal. The data model and metadata model are distinct to the point that there is no simple way to provide services to both data and metadata.

The state of XML and related technologies, while stabalizing, is still somewhat problematic. XML Schema has just become a full recommendation and COTS products are just now catching up to that recommendation. This conflict between versions of the candidatae recommendations has caused some difficulties in integrating different pars of the prototype, but he number of conflicts between different versions of standards is dropping considerably.

References:

[1] Siri J. S. Khalsa, S. Amer, H. Direskeneli, N. LaBelle-Hamer, A. K. Sharma, Glenn Shirtliffe, and R. Raskin. *An ECS Data Provider's Guide to Metadata,* DRAFT White Paper, http://ecsinfo.gsfc.nasa.gov/metadata/wpdraft.html, January, 1997.

[2] S. Lewick, K. Cream, S. Gluck, S. Paradise, A. Shaner, *Science Data Processing Interface Control Document,* Jet Propulsion Laboratory, California Institute of Technology, March, 1998.

[3]E. R. Harold, *XML Bible,* IDG Books Worldwide, Inc., New York, N.Y, 1999